245721

STIC EIC 2100
Search Request Form

Today's Date: 12/13/07

What date would you like to use to limit the search?

Priority Date: 01/08/99 Other:

Name Andy Ly

AU 2162 Examiner # 77831

Room # 3B39 Phone 2-4039

Serial # 09/226,939

Format for Search Results (Circle One):

PAPER DISK EMAIL

Where have you searched so far?

USP DWPI EPO JPO ACM IBM TDB

IEEE INSPEC SPI Other _____

Is this a "Fast & Focused" Search Request? (Circle One) YES NO

A "Fast & Focused" Search is completed in 2-3 hours (maximum). The search must be on a very specific topic and meet certain criteria. The criteria are posted in EIC2100 and on the EIC2100 NPL Web Page at http://ptoweb/patents/stic/stic-tc2100.htm.

What is the topic, novelty, motivation, utility, or other specific details defining the desired focus of this search? Please include the concepts, synonyms, keywords, acronyms, definitions, strategies, and anything else that helps to describe the topic. Please attach a copy of the abstract, background, brief summary, pertinent claims and any citations of relevant art you have found.

Is this request for a BOARD of APPEALS case? (Circle One) YES NO

Is this case a SPECIAL CASE? (Circle One) YES NO

Recursively querying a database for one or more dependencies of procedural code objects stored in the database and identifying one or more dependencies of procedural code objects stored in the database

Abstract

⇒ "Generating the complete dependency graph of a database code object" (SQL language of PL/SQL)

STIC Searcher BM/jms Phone 2-3528

Date picked up 12/13/07 Date Completed 12/13/07

Search and Information Resources Administration

```
Set     Items   Description
S1     300364   DATABASE? OR DATABANK? OR DATA()(BASE? OR BANK? OR FILE? OR
              REPOSITOR? OR WAREHOUSE?) OR DB OR RDB OR OODB OR ODBC OR DB-
             MS
S2        265   S1(7N)((RECURS? OR REPEAT? OR PROGRESSIV? OR REITERAT? OR -
             ITERAT? OR REPRIS? OR (RUN OR DO OR EXECUT?)()AGAIN OR REDO??? .
              OR RE()(DO OR DOING OR RUN? ? OR RUNNING OR EXECUT?))(5N)(IN-
             QUIR? OR ENQUER???? OR QUERY??? OR QUERIE? ? OR REQUEST? OR A-
             SK??? OR QUESTION? OR SEARCH?))
S3       4173   (COMPLET? OR FINISH??? OR END??? OR ENTIR? OR FULLY OR TOT-
             AL? OR RESULT?)(5N)(SUBROUTIN? OR EMULAT? OR SUBPROGRAM? OR C-
             OMPUTER?(2N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROUTI-
             NE? OR SUBROUTIN? INFORMATION? OR DATA))
S4        766   S3(3N)(CREAT? OR PRODUC? OR DEVELOP? OR ORIGINAT? OR MAKE?
             OR MAKING? OR MADE OR GENERAT?)
S5      15301   (LOGIC? OR DIRECTION? OR FUNCTION? ? OR RULE?? OR METHOD??
             OR PROCEDUR? OR FORMULA? OR STRATEG? OR INSTRUCTION?? OR EXPR-
             ESSION???)(2N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROU-
             TINE? OR SUBROUTIN?)(2N)(CLASS?? OR OBJECT? OR ENTIT?)
S6       1294   S5(5N)(MULTI OR PLURAL? OR MORE(2N)ONE OR SEVERAL OR MANY -
             OR VARIOUS OR VARIET?)
S7         64   S6(7N)(MONITOR? OR EXAMIN? OR DETECT? OR UNCOVER? OR REVEA-
             L? OR ASSESS? OR EVALUAT? OR INSPECT?)
S8        192   S6(7N)(DETERMIN? OR COMPAR? OR DISCERN? OR ASCERTAIN? OR A-
             NALY? OR IDENT? OR CHECK? OR VERIF? OR JUDG???)
S9          0   S2 AND S4 AND S7:S8
S10         1   S2 AND S4
S11         2   S2 AND S3
S12         1   S11 NOT S10
S13         0   S2 AND PROCEDUR?(2N)CODE? ?(2N)(CLASS? OR OBJECT? OR ENTIT-
             ?)
S14         0   S2 AND S7:S8
S15         3   S3 AND S7:S8
S16         3   S15 NOT S10:S12
S17      1935   (REPAIR? OR FIX OR FIXE? ? OR FIXING OR PATCH? OR RESTOR? -
             OR CORRECT? OR DEBUG? OR DE()(BUG? ? OR BUGG???) OR REBUIL?)(-
             3N)(STORE? ? OR STORAG? OR STORING OR MEMOR?)(3N)(CLASS? OR O-
             BJECT? OR ENTIT?)
S18         1   S2 AND S17
S19         0   S18 NOT S10:S16
S20         0   S1:S2 AND S17 AND S3:S4 AND S5:S8
S21         0   S17 AND S3:S4 AND S5:S8
S22        59   S17 AND S5:S8
S23         8   S22 AND S1
S24         8   S23 NOT S10:S16
S25       213   S1 AND S17
S26         1   S25 AND S2
S27         0   S25 AND S3:S4 AND S5:S8
S28         8   S25 AND S5:S8
S29         9   S26:S28
S30         0   S29 NOT (S10:S16 OR S18 OR S23:S24)
S31         3   S25 AND (PROCEDUR? OR FUNCTION? OR FORMULA? OR INSTRUCT?)(-
             2N)(CODE? ? OR SCRIPT? OR PROGRAM? OR ROUTIN?)(2N)(CLASS? OR -
             OBJECT? OR ENTIT?)
S32         0   S31 NOT S28:S29
File 350:Derwent WPIX 1963-2007/UD=200779
         (c) 2007 The Thomson Corporation
File 347:JAPIO Dec 1976-2007/Jun(Updated 070926)
         (c) 2007 JPO & JAPIO
```

10/69/1 **(Item 1 from file: 350)**
DIALOG(R)File 350:Derwent WPIX

0010304689 - Drawing available
WPI ACC NO: 2000-618567/200059
XRPX Acc No: N2000-458404
Automatic subroutine information generation for DBMS, involves applying set of recursive algorithm and source code passing to code object repeatedly until new subroutine is not added to generated subroutine data
Patent Assignee: CHERNY I (CHER-I); COMPUTER ASSOC THINK INC (COMP-N); VINCENT J K (VINC-I)
Inventor: CHEMY I; CHERNY I; VICENT J K; VINCENT J K
Patent Family (11 patents, 87 countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	
WO 2000041100	A1	20000713	WO 2000US276	A	20000106	200059	B
AU 200027204	A	20000724	AU 200027204	A	20000106	200059	E
US 20010049682	A1	20011206	US 1999226939	A	19990108	200203	E
BR 200007412	A	20020409	BR 20007412	A	20000106	200232	E
			WO 2000US276	A	20000106		
KR 2001108075	A	20011207	KR 2001708647	A	20010707	200236	E
EP 1208459	A1	20020529	EP 2000905546	A	20000106	200243	E
			WO 2000US276	A	20000106		
CN 1342292	A	20020327	CN 2000804484	A	20000106	200247	E
ZA 200105551	A	20020731	ZA 20015551	A	20010705	200271	E
JP 2002534742	W	20021015	JP 2000592758	A	20000106	200282	E
			WO 2000US276	A	20000106		
AU 2004222703	A1	20041118	AU 200027204	A	20000106	200505	NCE
			AU 2004222703	A	20041019		
IL 144106	A	20061031	IL 144106	A	20000106	200680	E

Priority Applications (no., kind, date): US 1999226939 A 19990108; AU 2004222703 A 20041019

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
WO 2000041100	A1	EN	39	9	
National Designated States,Original: AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG UZ VN YU ZA ZW					
Regional Designated States,Original: AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW NL OA PT SD SE SL SZ TZ UG ZW					
AU 200027204	A	EN			Based on OPI patent WO 2000041100
BR 200007412	A	PT			PCT Application WO 2000US276
					Based on OPI patent WO 2000041100
EP 1208459	A1	EN			PCT Application WO 2000US276
					Based on OPI patent WO 2000041100
Regional Designated States,Original: AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE					
ZA 200105551	A	EN	44		
JP 2002534742	W	JA	38		PCT Application WO 2000US276
					Based on OPI patent WO 2000041100
AU 2004222703	A1	EN			Division of application AU 200027204
IL 144106	A	EN			Based on OPI patent WO 2000041100

Alerting Abstract WO A1

NOVELTY - A **complete subroutine** information is **generated** by repeatedly applying recursive algorithm and source code passing until a new

subroutine is not added to subroutine information. The generated subroutine information does not contain details of subroutine which involve code objects during execution of data manipulation statement and which implement object oriented code objects in the database.

DESCRIPTION - The subroutine information is maintained in a data structure termed as tracking array which is scanned to compile code objects in debug mode. Identification of calling paths in database code coverage tool, object profiling tool and object testing tool, identification of cyclic subroutines, database code objects and subroutine information presentation tool is done with subroutine information of subprograms in data management system.

An INDEPENDENT CLAIM is also included for A computer program product embedded on a computer readable medium for use in debugging a target data base code object.

USE - For **generating complete subroutine** information for relational database management system (RDBMS).

ADVANTAGE - Complexity in debugging and testing of code object is reduced and hence the time consumption and cost for debugging is reduced. The generated subroutine information does not contain code objects during execution of data manipulation statement.

DESCRIPTION OF DRAWINGS - The figure shows the flowchart of the method of **generating complete subroutine** information in DBMS.

Title Terms/Index Terms/Additional Words: AUTOMATIC; SUBROUTINES; INFORMATION; GENERATE; APPLY; SET; RECURSIVE; ALGORITHM; SOURCE; CODE; PASS; OBJECT; REPEAT; NEW; ADD; DATA

Class Codes

International Classification (Main): G06F-012/00, G06F-017/30
(Additional/Secondary): G06F-011/28, G06F-011/36
International Classification (+ Attributes)

IPC + Level Value Position Status Version					
G06F-0011/28	A	I	F	R	20060101
G06F-0011/36	A	I	L	R	20060101
G06F-0012/00	A	I	L	R	20060101
G06F-0017/30	A	I	L	R	20060101
G06F-0007/00	A	I		R	20060101
G06F-0009/44	A	I		R	20060101
G06F-0011/36	A	I	L		20060101
G06F-0017/30	A	I	L		20060101
G06F-0017/30	A	I		R	20060101
G06F-0003/06	A	I	F		20060101
G06F	S	I		R	20060101
G06F-0011/28	C	I	F	R	20060101
G06F-0011/36	C	I	L	R	20060101
G06F-0012/00	C	I	L	R	20060101
G06F-0017/30	C	I	L	R	20060101
G06F-0017/30	C	I		R	20060101
G06F-0007/00	C	I		R	20060101
G06F-0009/44	C	I		R	20060101

US Classification, Issued: 707100000
File Segment: EPI;
DWPI Class: T01
Manual Codes (EPI/S-X): T01-J05B3; T01-J05B4B; T01-J05B4M; T01-S03
?

16/69,K/3 (Item 3 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0005801658 - Drawing available
WPI ACC NO: 1992-024591/199203
Related WPI Acc No: 1991-223082; 1992-024596; 1992-024599; 1992-024602; 1992-024601; 1992-268838; 1992-097074; 1992-097065; 1992-097064; 1992-097062; 1992-097060; 1992-024605; 1992-024604; 1992-024603; 1992-024598; 1992-024593; 1992-024589; 1992-024590; 1992-024587
XRPX Acc No: N1992-018741
Instruction scheduling optimisation on processor - rearranging machine instructions into order that will result in fastest execution based upon resource interaction simulation
Patent Assignee: CRAY RES INC (CRAY-N); SUPERCOMPUTER SYSTEMS LP (SUPE-N)
Inventor: RASBOLD J C; VAN DYKE D A; VANDYKE D A
Patent Family (10 patents, 16 countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	
WO 1991020031	A	19911226	WO 1991US4073	A	19910610	199203	B
EP 535107	A1	19930407	EP 1991911933	A	19910610	199314	E
			WO 1991US4073	A	19910610		
US 5202975	A	19930413	US 1990537466	A	19900611	199317	E
			US 1990571500	A	19900823		
			US 1992896895	A	19920610		
JP 5508040	W	19931111	JP 1991511254	A	19910610	199350	E
			WO 1991US4073	A	19910610		
TW 213998	A	19931001	TW 1991109901	A	19911218	199351	E
US 5307478	A	19940426	US 1990537466	A	19900611	199416	NCE
			US 1992896895	A	19920610		
			US 1992969789	A	19921029		
TW 237529	A	19950101	TW 1991104744	A	19911218	199511	E
EP 535107	B1	19991208	EP 1991911933	A	19910610	200002	E
			WO 1991US4073	A	19910610		
DE 69131830	E	20000113	DE 69131830	A	19910610	200010	E
			EP 1991911933	A	19910610		
			WO 1991US4073	A	19910610		
JP 3288372	B2	20020604	JP 1991511254	A	19910610	200240	E
			WO 1991US4073	A	19910610		

Priority Applications (no., kind, date): US 1992969789 A 19921029; US 1992896895 A 19920610; US 1990537466 A 19900611; US 1990571500 A 19900823

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
WO 1991020031	A	EN			
National Designated States,Original: JP KR					
Regional Designated States,Original: AT BE CH DE DK ES FR GB GR IT LU NL SE					
EP 535107	A1	EN			PCT Application WO 1991US4073 Based on OPI patent WO 1991020031
Regional Designated States,Original: DE FR GB					
US 5202975	A	EN	13	5	C-I-P of application US 1990537466 Continuation of application US 1990571500 C-I-P of patent US 5179702
JP 5508040	W	JA			PCT Application WO 1991US4073 Based on OPI patent WO 1991020031
TW 213998	A	ZH			
US 5307478	A	EN	13	5	C-I-P of application US 1990537466

				Division of application US 1992896895
				C-I-P of patent US 5179702
				Division of patent US 5202975
TW 237529	A	ZH		
EP 535107	B1	EN		PCT Application WO 1991US4073
				Based on OPI patent WO 1991020031
Regional Designated States,Original:				DE FR GB
DE 69131830	E	DE		Application EP 1991911933
				PCT Application WO 1991US4073
				Based on OPI patent EP 535107
				Based on OPI patent WO 1991020031
JP 3288372	B2	JA	14	PCT Application WO 1991US4073
				Previously issued patent JP 05508040
				Based on OPI patent WO 1991020031

Alerting Abstract WO A

The method schedules instructions for a processor having multiple functional resources where the reordering is accomplished in response to a simulation of the run-time environment of the target machine. The simulation of the run-time environment is performed at compile time, after the machine instructions have been generated by a compiler, or after instruction generation by an assembler.

The machine instructions for a basic block of instructions are rearranged into an order that will result in the faster execution based upon the results of the simulation of the interaction of the multiple functional resources in the target machine.

USE/ADVANTAGE - Scheduling of machine instructions to multiple functional resource in processor during software compilation of source code. Maximises instruction flow. Minimises pipeline halts. @(19pp Dwg.No.3/5)@

Equivalent Alerting Abstract US A

The method for scheduling instructions for a processor having multiple functional resources includes reordering the instructions in response to a simulation of the run-time environment of the target machine. The simulation of the run-time environment of the target machine is performed at compile time after the machine instructions have been generated by a compiler, or after instruction generation by an assembler.

The method involves rearranging the machine instructions for a basic block of instructions into an order that will result in the fastest execution based upon the results of the simulation of the interaction of the multiple functional resources in the target machine.

USE/ADVANTAGE - Rearranging order in which machine instructions within basic block of instructions are issued to processor contg. multiple functional resources, reduces overall execution time of basic block of instructions.

Equivalent Alerting Abstract US A

During the compilation or assembly of the source code program to produce the number of object code instructions, an optimum path of execution for the object code instruction is determined among all available functional units in the target computer in which the object code instruction could execute. Various sequences of execution of a number of object code instructions are simulated within a basic block of object code instructions that could be executed next for all combinations of available functional units in the target computer. An sequence of execution of the number of object code instructions is selected as an optimum order of execution which results in the fastest execution speed during the simulation step.

For each object code instruction in the number of object code instructions, an optimum path of execution is assigned among all available

functional units in the target computer in which the object code instruction could execute based on the optimum order of execution. If the optimum path of execution for the object code instruction is different than a preselected default path of execution for an object code instruction for a similar type of object code operation, a path instruction is inserted prior to the execution of the object code instruction directing that the next object code instruction be executed in the target computer according to the optimum path of execution for the object code instruction.

USE/ADVANTAGE - For inserting path instruction during compilation of computer programs for processor having multiple functional resources. Reduces overall execution time of basic block of instructions.

Title Terms/Index Terms/Additional Words: INSTRUCTION; SCHEDULE; OPTIMUM; PROCESSOR; REARRANGE; MACHINE; ORDER; RESULT; FAST; EXECUTE; BASED; RESOURCE; INTERACT; SIMULATE

Class Codes

International Classification (Main): G06F-015/00, G06F-009/45
(Additional/Secondary): G06F-009/30, G06F-009/38, G06F-009/455
US Classification, Issued: 395500000, 395375000, 395700000, 395800000, 395500000, 395700000, 364221200, 364280500, 364916300, 364973000

File Segment: EPI;
DWPI Class: T01
Manual Codes (EPI/S-X): T01-F03; T01-F05

Original Publication Data by Authority

Claims:

...multiple functional units capable of performing similar types of object code operations, a method of **determining** in which of a **plurality of the** functional units that **an object code instruction should** execute. the **method** comprising the computer-implemented steps of: during the compilation or assembly of the **source** code program to produce the plurality of object code instructions, determining an optimum path of...

...code instruction among all available functional units in the target computer in which the object **code instruction** could execute **by performing** the steps of simulating various **sequences of execution** of a plurality of **object code instructions** within **a basic block** of **object code instructions** that could be executed next for all combinations of available functional units in the target...

...object code instructions, assigning an optimum path of execution among all available functional units in **the** target **computer in** which the object code instruction could execute based on the optimum order of execution; and...

?

24/69,K/7 (Item 7 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0010322413 - Drawing available
WPI ACC NO: 2000-636931/200061
XRPX Acc No: N2000-472229
Dynamic debugging information generation for computer-aided software engineering, by compiling source code of component, to generate new symbolic debugging data, when symbolic debugging information is invalid
Patent Assignee: OBJECT TECHNOLOGY LICENSING CORP (OBJE-N)
Inventor: MCINERNEY P J; WIMBLE M D; YOU L L
Patent Family (1 patents, 1 countries)

Patent			Application			
Number	Kind	Date	Number	Kind	Date	Update
US 6067641	A	20000523	US 1995557767	A	19951113	200061 B
			US 1999246789	A	19990209	

Priority Applications (no., kind, date): US 1995557767 A 19951113; US 1999246789 A 19990209

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
US 6067641	A	EN	63	42	Continuation of application US 1995557767
					Continuation of patent US 5956479

Alerting Abstract US A
NOVELTY - The program counter value is used to locate a component in the memory, when program execution halts during debugging. The located component object code is related with its corresponding source code, to ascertain the validity of debugging information. When debugging information is not valid, the source code of the located component is compiled to generate new symbolic debugging information.
DESCRIPTION - The name components in the memory are stored and retrieved from a **database** . A compiler compiles the source code of each component to obtain object code. The compilation of components is sequenced responding to the component properties and dependencies.
USE - For demand-based generation of symbolic debugger information for computer-aided software engineering.
ADVANTAGE - Since information required for debug operation is built rather than the entire program, incremented debugging capability is improved.
DESCRIPTION OF DRAWINGS - The figure showing the block diagram of computer system.

Title Terms/Index Terms/Additional Words: DYNAMIC; DEBUG; INFORMATION; GENERATE; COMPUTER; AID; SOFTWARE; ENGINEERING; COMPILE; SOURCE; CODE; COMPONENT; NEW; SYMBOL; DATA; INVALID

Class Codes
International Classification (+ Attributes)
IPC + Level Value Position Status Version
G06F-0011/36 A I R 20060101
G06F-0011/36 C I R 20060101
US Classification, Issued: 714038000, 712227000

File Segment: EPI;
DWPI Class: T01
Manual Codes (EPI/S-X): T01-F05A; T01-J20C

Alerting Abstract DESCRIPTION - The name components in the memory are stored and retrieved from a **database** . A compiler compiles the source code of each component to obtain object code. The compilation...

Original Publication Data by Authority

Original Abstracts:
...a function. One major functionality built in HOOPS is the debugger, using symbolic properties. The **database** stores the components and properties. The debugger, using a GUI, displays to the user the...

...code and retrieves source code configuration as needed. Symbolic properties that are stored in the **database** can be removed to reduce **database** and disk memory usage; they can be later reconstructed using the same method of demand...

Claims:
...each component including an attribute representing the validity of the component data, source code for **implementing** properties of the component and **object code for executing** the component, the **method** comprising the steps of: (a) providing class libraries for retention in the memory from which may be instantiated (1) a **database** functionally for persistently storing and **retrieving** the named components in the memory, (2) a compiler functionality for calculating dependencies associated with ...

...and dependencies: (b) providing a run-time environment to (1) support the instantiation of the **database** , compiler and build functionalities; and (2) support the execution of the debugger and **the** program such that (i) when program execution halts during debugging, using the program counter value...

...memory, (ii) checking the attribute of the located component to ascertain the validity of symbolic **debugging** information relating **the** located component **object** code to the located component source code, (iii) generating new symbolic debugging information by compiling...

24/69,K/8 (Item 8 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0007306471 - Drawing available
WPI ACC NO: 1995-368079/199548
Related WPI Acc No: 1995-368080
XRPX Acc No: N1995-272421
Software program object distribution between producer and potential user - loading software object which is encrypted onto computer-accessible memory media along with file management program
Patent Assignee: IBM CORP (IBMC); INT BUSINESS MACHINES CORP (IBMC)
Inventor: COOPER T E; PHILIPS H W; PRYOR R F
Patent Family (12 patents, 9 countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update	
EP 679979	A1	19951102	EP 1995105442	A	19950411	199548	B
AU 199514856	A	19951102	AU 199514856	A	19950315	199551	E
JP 7295803	A	19951110	JP 199590472	A	19950417	199603	E
BR 199501522	A	19951121	BR 19951522	A	19950410	199604	E
CA 2145925	A	19951026	CA 2145925	A	19950330	199610	E
CN 1115059	A	19960117	CN 1995104201	A	19950414	199740	E
US 5689560	A	19971118	US 1994235035	A	19940425	199801	E
CA 2145925	C	19981208	CA 2145925	A	19950330	199908	E
KR 200444	B1	19990615	KR 19959726	A	19950425	200060	E
EP 679979	B1	20030618	EP 1995105442	A	19950411	200341	E
DE 69531077	E	20030724	DE 69531077	A	19950411	200356	E
			EP 1995105442	A	19950411		
CN 1132110	C	20031224	CN 1995104201	A	19950414	200564	E

Priority Applications (no., kind, date): US 1994235035 A 19940425; US 1994235032 A 19940425; EP 1995105442 A 19950411

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
EP 679979	A1	EN	49	35	
Regional Designated States,Original: DE FR GB					
JP 7295803	A	JA	1		
BR 199501522	A	PT			
CA 2145925	A	EN			
US 5689560	A	EN	44	35	
CA 2145925	C	EN			
EP 679979	B1	EN			
Regional Designated States,Original: DE FR GB					
DE 69531077	E	DE			Application EP 1995105442 Based on OPI patent EP 679979

Alerting Abstract EP A1
A software object, a computer-accessible memory media and a file management program are provided. The software object is reversibly functionally limited by encryption. The object is recorded onto the media which is shipped from the producer to the potential user.
The file management program is loaded into a user-controlled data processing system.
It associates with an operating system for the system. The program is executed with the system to restrict access to the software object.
ADVANTAGE - Enables software trial with try-and-buy user interaction.

Title Terms/Index Terms/Additional Words: SOFTWARE; PROGRAM; OBJECT; DISTRIBUTE; PRODUCE; POTENTIAL; USER; LOAD; ENCRYPTION; COMPUTER; ACCESS; MEMORY; MEDIUM; FILE; MANAGEMENT

Class Codes
International Classification (Main): G06F-017/60
International Classification (+ Attributes)

IPC + Level	Value	Position		Status	Version
G06F-0001/00	A	I		R	20060101
G06F-0012/14	A	I	F	R	20060101
G06F-0021/00	A	I		R	20060101
G06F-0021/22	A	I	L	R	20060101
G06F-0021/24	A	I	L	R	20060101
G09C-0001/00	A	I	L	R	20060101
G06F-0001/00	C	I		R	20060101
G06F-0012/14	C	I	F	R	20060101
G06F-0021/00	C	I		R	20060101
G06F-0021/22	C	I	L	R	20060101
G09C-0001/00	C	I	L	R	20060101

US Classification, Issued: 380004000, 380009000, 380023000, 380025000, 380049000, 380050000, 340825310, 340825340, 395726000

File Segment: EPI;
DWPI Class: T01
Manual Codes (EPI/S-X): T01-H07C; T01-J12B; T01-J12C; T01-J20B

Original Publication Data by Authority

Original Abstracts:
...The computer-accessible memory media is read with the user-controlled data processing system. The **file** management program is utilized to restrict access to the software object...
...producer to the potential user. The file management program is loaded into a user-controlled **data** processing system, and associated with the operating system for the user-controlled data processing system...
Claims:
... A **method** of distributing software **objects** from a producer to a potential user, comprising the method steps of:providing a software...

...reading said computer-accessible memory media with said user-controlled data processing system;utilizing said **file** management **program** to **restore** said **function** **of** said software **object** with said user-controlled data processing system to allow access to said software object following

```
Set     Items   Description
S1     281670   DATABASE? OR DATABANK? OR DATA()(BASE? OR BANK? OR FILE? OR
              REPOSITOR? OR WAREHOUSE?) OR DB OR RDB OR OODB OR ODBC OR DB-
             MS OR RDBMS
S2        524   S1(7N)((RECURS? OR REPEAT? OR PROGRESSIV? OR REITERAT? OR -
             ITERAT? OR REPRIS? OR (RUN OR DO OR EXECUT?)()AGAIN OR REDO???
              OR RE()(DO OR DOING OR RUN? ? OR RUNNING OR EXECUT?))(5N)(IN-
             QUIR? OR ENQUER???? OR QUERY??? OR QUERIE? ? OR REQUEST? OR A-
             SK??? OR QUESTION? OR SEARCH?))
S3       9777   (COMPLET? OR FINISH??? OR END??? OR ENTIR? OR FULLY OR TOT-
             AL? OR RESULT?)(5N)(SUBROUTIN? OR EMULAT? OR SUBPROGRAM? OR C-
             OMPUTER?(2N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROUTI-
             NE? OR SUBROUTIN? OR INFORMATION? OR DATA))
S4       1190   S3(5N)(CREAT? OR PRODUC? OR DEVELOP? OR ORIGINAT? OR MAKE?
             OR MAKING? OR MADE OR GENERAT?)
S5       9718   (LOGIC? OR DIRECTION? OR FUNCTION? ? OR RULE?? OR METHOD??
             OR PROCEDUR? OR FORMULA? OR STRATEG? OR INSTRUCTION?? OR EXPR-
             ESSION???)(3N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROU-
             TINE? OR SUBROUTIN?)(3N)(CLASS?? OR OBJECT? OR ENTIT?)
S6        925   S5(5N)(MULTI OR PLURAL? OR MORE(2N)ONE OR SEVERAL OR MANY -
             OR VARIOUS OR VARIET?)
S7         54   S6(7N)(MONITOR? OR EXAMIN? OR DETECT? OR UNCOVER? OR REVEA-
             L? OR ASSESS? OR EVALUAT? OR INSPECT?)
S8         96   S6(7N)(DETERMIN? OR COMPAR? OR DISCERN? OR ASCERTAIN? OR A-
             NALY? OR IDENT? OR CHECK? OR VERIF? OR JUDG???)
S9          0   S2(100N)S4(100N)S5
S10         0   S2(100N)S3(100N)S5
S11       865   (REPAIR? OR FIX OR FIXE? ? OR FIXING OR PATCH? OR RESTOR? -
             OR CORRECT? OR DEBUG? OR DE()(BUG? ? OR BUGG???) OR REBUIL?)(-
             3N)(STORE? ? OR STORAG? OR STORING OR MEMOR?)(3N)(CLASS? OR O-
             BJECT? OR ENTIT?)
S12         0   S2(100N)S11
S13       131   S1:S2(100N)S11
S14         1   S13(100N)RECURS?
S15         2   S13(100N)S3:S4(100N)S5:S8
S16         2   S15 NOT S14
S17         3   S11(100N)S3:S4(100N)S5:S8
S18         3   S16:S17
S19        12   S13(100N)(DEBUG????)
S20        11   S19 NOT S14:S18
File 348:EUROPEAN PATENTS 1978-2007/ 200750
        (c) 2007 European Patent Office
File 349:PCT FULLTEXT 1979-2007/UB=20071129UT=20071122
        (c) 2007 WIPO/Thomson
```

14/5,K/1 **(Item 1 from file: 349)**
DIALOG(R)File 349:PCT FULLTEXT

00577727 **Image available**
SYSTEM AND METHOD FOR RECURSIVE PATH ANALYSIS OF DBMS PROCEDURES
SYSTEME ET PROCEDE D'ANALYSE RECURSIVE DE PROCEDURES DE SYSTEMES DE GESTION DE BASE DE DONNEES (SGBD)

Patent Applicant/Assignee:
COMPUTER ASSOCIATES THINK INC,
Inventor(s):
VINCENT John K,
CHERNY Igor,

Patent and Priority Information (Country, Number, Date):
Patent: WO 200041100 A1 20000713 (WO 0041100)
Application: WO 2000US276 20000106 (PCT/WO US0000276)
Priority Application: US 99226939 19990108
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG UZ VN YU ZA ZW GH GM KE LS MW SD SL SZ TZ UG ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG
Main International Patent Class (v7): G06F-017/30
Publication Language: English
Fulltext Availability:
Detailed Description
Claims
Fulltext Word Count: 7796

English Abstract
A system, method and **database** development tools are disclosed for automatically generating the complete dependency graph (12) for use in **debugging** **stored** code **objects** (11) in a **database** , by using a **recursive** dependency tracking algorithm (14) which takes into consideration the indirect dependencies (15) on triggers as well as the dependencies on implementations of object oriented code objects which are represented as separate objects in the **database** catalog.

French Abstract
La presente invention concerne un systeme, un procede et un instrument de mise au point de bases de donnees, visant a generer automatiquement le graphe complet (12) des dependances afin d'effectuer la mise au point d'objets (11) a code memorises dans une base de donnees. Ledit systeme utilise un algorithme (14) recursif de recherche de dependances, tenant compte des dependances indirectes (15) des declenchements ainsi que des dependances de la mise en oeuvre d'objets a code orientes-objet, qui sont representes comme des objets separes dans le catalogue de la base de donnees.

Fulltext Availability:
Detailed Description

English Abstract
A system, method and **database** development tools are disclosed for automatically generating the complete dependency graph (12) for use in **debugging** **stored** code **objects** (11) in a **database** , by using a

recursive dependency tracking algorithm (14) which takes into consideration the indirect dependencies (15) on triggers as...

...on implementations of object oriented code objects which are represented as separate objects in the **database** catalog.

Detailed Description

... The solution to this technical problem developed by applicants uses a query that is called **recursively** . An array is used to track the parents so that the graph can be reconstructed...

...it is determined whether the dependency already occurs in the graph. If it occurs, the **recursion** is stopped.

DISCLOSURE OF THE INVENTION

A system, method and **database** development tool are disclosed for automatically generating the complete dependencies of a stored code object in a **database** by applying a set of **recursive** procedures and parsing the source code of the code objects.

Also a method for generating...

...and their relationship with one another is claimed.

Additionally claimed are a method for generating **debug** versions of **stored** code **objects** and all its dependencies. Also claimed is a method for identifying potential run-time errors...

...dependent code objects. Also claimed is a method for identifying the cyclic dependencies of a **database** code object. Also claimed is a method of debugging code objects in a **database** using the complete dependency graph of the particular code object. Also claimed is a method of developing **database** programs comprising a computer system and a program code mechanism for automatically generating complete dependencies...

?

```
Set     Items   Description
S1    1230434   DATABASE? OR DATABANK? OR DATA()(BASE? OR BANK? OR FILE? OR
            REPOSITOR? OR WAREHOUSE?) OR DB OR RDB OR OODB OR ODBC OR DB-
            MS OR RDBMS
S2       1217   S1(7N)((RECURS? OR REPEAT? OR PROGRESSIV? OR REITERAT? OR -
            ITERAT? OR REPRIS? OR (RUN OR DO OR EXECUT?)()AGAIN OR REDO???
             OR RE()(DO OR DOING OR RUN? ? OR RUNNING OR EXECUT?))(5N)(IN-
            QUIR? OR ENQUER???? OR QUERY??? OR QUERIE? ? OR REQUEST? OR A-
            SK??? OR QUESTION? OR SEARCH?))
S3      28865   (COMPLET? OR FINISH??? OR END??? OR ENTIR? OR FULLY OR TOT-
            AL? OR RESULT?)(5N)(SUBROUTIN? OR EMULAT? OR SUBPROGRAM? OR C-
            OMPUTER?(2N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROUTI-
            NE? OR SUBROUTIN? OR INFORMATION? OR DATA))
S4       3455   S3(5N)(CREAT? OR PRODUC? OR DEVELOP? OR ORIGINAT? OR MAKE?
            OR MAKING? OR MADE OR GENERAT?)
S5      28021   (LOGIC? OR DIRECTION? OR FUNCTION? ? OR RULE?? OR METHOD??
            OR PROCEDUR? OR FORMULA? OR STRATEG? OR INSTRUCTION?? OR EXPR-
            ESSION???)(3N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROU-
            TINE? OR SUBROUTIN?)(3N)(CLASS?? OR OBJECT? OR ENTIT?)
S6       2788   S5(5N)(MULTI OR PLURAL? OR MORE(2N)ONE OR SEVERAL OR MANY -
            OR VARIOUS OR VARIET?)
S7         80   S6(7N)(MONITOR? OR EXAMIN? OR DETECT? OR UNCOVER? OR REVEA-
            L? OR ASSESS? OR EVALUAT? OR INSPECT?)
S8        177   S6(7N)(DETERMIN? OR COMPAR? OR DISCERN? OR ASCERTAIN? OR A-
            NALY? OR IDENT? OR CHECK? OR VERIF? OR JUDG???)
S9          0   S2 AND S4
S10         1   S2 AND S3
S11        93   S2 AND S5:S8
S12         0   S11 AND S7:S8
S13         0   S11 AND S3
S14        93   S11 AND RECURS?
S15        92   S14 AND PY=1978:1999
S16       253   (REPAIR? OR FIX OR FIXE? ? OR FIXING OR PATCH? OR RESTOR? -
            OR CORRECT? OR DEBUG? OR DE()(BUG? ? OR BUGG???) OR REBUIL?)(-
            3N)(STORE? ? OR STORAG? OR STORING OR MEMOR?)(3N)(CLASS? OR O-
            BJECT? OR ENTIT?)
S17         0   S15 AND S16
S18         0   S16 AND S2
S19         2   S15 AND DEBUG?
S20         0   S16 AND S4
S21        90   S15 NOT S19
S22         0   S21 AND S4
S23        90   S21 AND S5:S8
S24        90   S23 AND RECURS?(5N)QUER????
S25        88   RD  (unique items)
S26         0   S25 AND S3
S27         6   S25 AND (SUBROUTIN? OR EMULAT? OR SUBPROGRAM? OR COMPUTER?-
            (2N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROUTINE? OR S-
            UBROUTIN? OR INFORMATION? OR DATA))
S28        24   S16 AND S1
S29         0   S28 AND S5:S8
S30         3   S28 AND (SUBROUTIN? OR EMULAT? OR SUBPROGRAM? OR COMPUTER?-
            (2N)(CODE? OR UTILIT? OR SCRIPT? OR PROGRAM? OR ROUTINE? OR S-
            UBROUTIN? OR INFORMATION? OR DATA))
S31         3   S30 NOT S27
S32        21   S28 NOT (S10 OR S19 OR S27 OR S30)
S33        13   RD  (unique items)
File   2:INSPEC 1898-2007/Nov W4
         (c) 2007 Institution of Electrical Engineers
File   6:NTIS 1964-2007/Dec W4
         (c) 2007 NTIS, Intl Cpyrght All Rights Res
```

Abstract Npl. files

File 8:Ei Compendex(R) 1884-2007/Dec W1
(c) 2007 Elsevier Eng. Info. Inc.
File 34:SciSearch(R) Cited Ref Sci 1990-2007/Dec W2
(c) 2007 The Thomson Corp
File 35:Dissertation Abs Online 1861-2007/Aug
(c) 2007 ProQuest Info&Learning
File 56:Computer and Information Systems Abstracts 1966-2007/Oct
(c) 2007 CSA.
File 60:ANTE: Abstracts in New Tech & Engineer 1966-2007/Nov
(c) 2007 CSA.
File 62:SPIN(R) 1975-2007/Nov W4
(c) 2007 American Institute of Physics
File 65:Inside Conferences 1993-2007/Dec 12
(c) 2007 BLDSC all rts. reserv.
File 95:TEME-Technology & Management 1989-2007/Dec W2
(c) 2007 FIZ TECHNIK
File 99:Wilson Appl. Sci & Tech Abs 1983-2007/Oct
(c) 2007 The HW Wilson Co.
File 111:TGG Natl.Newspaper Index(SM) 1979-2007/Nov 30
(c) 2007 The Gale Group
File 144:Pascal 1973-2007/Dec W1
(c) 2007 INIST/CNRS
File 239:Mathsci 1940-2007/Dec
(c) 2007 American Mathematical Society
File 256:TecInfoSource 82-2007/Apr
(c) 2007 Info.Sources Inc
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
(c) 2006 The Thomson Corp
File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13
(c) 2002 The Gale Group

27/7/1 (Item 1 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

07623932 E.I. No: EIP97023518495
Title: Recursive query **processing using graph traversal techniques**
Author: Pulido, Estrella
Corporate Source: Univ of Bristol, Bristol, UK
Conference Title: Proceedings of the 1996 5th ACM CIKM International Conference on Information and Knowledge Management
Conference Location: Rockville, MD, USA Conference Date: 19961112-19961116
Sponsor: ACM; SIGIR; SIGART
E.I. Conference No.: 45973
Source: International Conference on Information and Knowledge Management, Proceedings 1996.. p 37-44
Publication Year: 1996
CODEN: 002176
Language: English
Document Type: CA; (Conference Article) Treatment: G; (General Review); T; (Theoretical)
Journal Announcement: 9704W1
Abstract: This paper presents STARBASE, a new algorithm for **recursive query** processing on deductive **databases** base on the Chart Parsing algorithm. It is distinct from the other applications of parsing to deduction, namely Earley Deduction and Rosenblueth's method, because it removes variables from literals and extends the Chart Parsing engine to handle all possible variations in the pattern of arguments in the literals of deduction rules. Like other tabling methods, STARBASE avoids redundant computation by storing and reusing partial results but, in contrast with them, it does not depend on subsumption and uses syntactic equality checking, instead. Because STARBASE takes a strongly graph-oriented view of both the database and the deduction rules, the evaluation of a query on a database can be viewed as a process of traversing paths in the graph representing the database. A prototype of the STARBASE system has been implemented in the C language. Performance results show that STARBASE, even in prototype form, lies within the performance range of the most advanced existing systems. (Author abstract) 13 Refs.

27/7/2 (Item 2 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

05804105 E.I. Monthly No: EIM8910-033325
Title: Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.
Author: Anon.
Conference Title: Proceedings of the Eight ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems
Conference Location: Philadelphia, PA, USA Conference Date: 19890329
Sponsor: Special Interest Group for Automata and Computability Theory; Special Interest Group for the Management of Data; Special Interest Group for Artificial Intelligence
E.I. Conference No.: 12323
Source: Proc Eighth ACM SIGACT-SIGMOD-SIGART Symp Princ Database Syst 1989. Publ by ACM, New York, NY, USA. 401p
Publication Year: 1989
ISBN: 0-89791-308-6
Language: English
Document Type: CP; (Conference Proceedings) Treatment: A; (Applications); T; (Theoretical)
Journal Announcement: 8910
Abstract: This conference proceedings contains 38 papers. The main subjects are logic programs and programming, application of logic programming to databases, query processing complexity, query evaluation algorithms, Horn Tables, handling of incomplete information in database, automata theory, **database** updating, testing of normal forms, **recursive query** processing, modular architectures for distributed and **database** systems, clustered multiattribute hash files, B-trees, secondary key retrieval, declustering using error correcting codes, concurrency control, query languages, object-oriented databases, and **logic** for **object** -oriented **logic programming** .

27/7/3 **(Item 1 from file: 34)**
DIALOG(R)File 34:SciSearch(R) Cited Ref Sci

03459376 Genuine Article#: NE818 Number of References: 12
Title: ADVANCED INTELLIGENT NETWORK SERVICE LABORATORY
Author(s): BOSCO PG; FARACI F
Corporate Source: CSELT SPA,VIA REISS ROMOLI 274/I-10148 TURIN//ITALY/
Journal: CANADIAN JOURNAL OF ELECTRICAL AND COMPUTER ENGINEERING-REVUE CANADIENNE DE GENIE ELECTRIQUE ET INFORMATIQUE, 1994, V19, N1 (JAN) , P21-25
ISSN: 0840-8688
Language: ENGLISH Document Type: ARTICLE
Abstract: This paper presents the architecture of a laboratory for design, validation and execution of advanced intelligent network (AIN) services. The AIN service laboratory permits testing of new services from design to execution. It is composed of two main parts: a design environment, where design and logical validation take place, and a hardware/software system closely **emulating** a real AIN structure (with service switching points, service control points, etc.), where the service can be directly tested. A technical overview of the AIN service laboratory is provided, describing the following issues: a formal and practical approach to validation, ranging from traditional human inspection to automatic verification of global temporal properties; a logic-programming-based concurrency model and nondeterministic backtrackable simulator on which SDL processes, representing IN functional entities and agents, are mapped; animation of network scenarios; a service logic execution environment designed on the basis of IN functional architecture defined in CCITT draft recommendation Q1214; a standard basic call state model (BCSM)-based call control, implemented in a service switching point (SSP) **emulator** . Furthermore, a description of an advanced universal personal telecommunication (UPT) service for which new network capabilities such as speech recognition are integrated and tested in the AIN service laboratory, is provided as an example.

27/7/4 (Item 2 from file: 34)
DIALOG(R)File 34:SciSearch(R) Cited Ref Sci

02902926 Genuine Article#: MP441 Number of References: 18
Title: SCHEDULING SPECULATIVE WORK IN MUSE AND PERFORMANCE RESULTS
Author(s): ALI KAM; KARLSSON R
Corporate Source: SICS/S-16428 KISTA//SWEDEN/
Journal: INTERNATIONAL JOURNAL OF PARALLEL PROGRAMMING, **1992** , V21, N6 (DEC), P449-476
ISSN: 0885-7458
Language: ENGLISH Document Type: ARTICLE
Abstract: Work which may later be pruned is called speculative work. In this paper we present and evaluate a simple and efficient strategy, used in the Muse OR-parallel Prolog system, for better scheduling of speculative work. The strategy concentrates workers on the leftmost available work in the Prolog tree as long as there exists enough parallelism, thus **emulating** the sequential Prolog execution as much as possible. This strategy therefore makes it less probable that unnecessary work is executed. A new cut scheme that reduces the amount of speculative work is also presented. The performance results of our strategy are compared with the performance results of similar strategies implemented in the Aurora OR-parallel Prolog system. The comparison shows that our strategy performs quite well.

27/7/5 (Item 1 from file: 56)
DIALOG(R)File 56:Computer and Information Systems Abstracts

0000216091 IP ACCESSION NO: 0031265

Network-based simple recursive **answer evaluation for deductive databases in parallel environment.**

Kim, Keecheon; Henschen, L J
Northwestern Univ, Evanston, IL, USA

ADDL. SOURCE INFO: PROC 4 INT CONF SOFTWARE ENG KNOWLEDGE ENG., IEEE, COMPUTER SOCIETY, LOS ALAMITOS, CA (USA), 1992, pp. 63-70,
PUBLICATION DATE: 1992

PUBLISHER: IEEE, COMPUTER SOCIETY, LOS ALAMITOS, CA (USA)

CONFERENCE:
the 4th International Conference on Software Engineering and Knowledge Engineering, Capri, Italy, 06/15-20/92

RECORD TYPE: Abstract
LANGUAGE: English
ISBN: 0-8186-2830-8
FILE SEGMENT: Computer & Information Systems Abstracts

ABSTRACT:
Because of increasing needs and requirements for the use of databases, we always look for more efficient ways to handle database access. Parallel computing environments draw more attention to achieve the high processing speed and the less expensive processing method. In this paper, we propose the possibility of using a connectionist model by treating every datum as an active processing unit cooperating with other such units via messages in getting answers in a deductive **database** , especially with **recursion** . We use **query** compilation and **iteration** to process the **recursive** case. The notion of the reverse-compilation which is essential to get the correct answers is introduced in handling **recursive** cases. We introduce the **recursive** controller which is dedicated to processing the **recursive** cases as a sublayer to a central database controller.

27/7/6 **(Item 1 from file: 144)**
DIALOG(R)File 144:Pascal

13790243 PASCAL No.: 98-0504567
Querying sequence databases with transducers
DBPL-6 : database programming languages : Estes Park CO, 18-20 August 1997
BONNER A J; MECCA G
CLUET Sophie, ed; HULL Rick, ed
University of Toronto - Department of Computer Science, Toronto, ON, Canada; D.I.F.A. - Universita della Basilicata - via della Tecnica, 3, 85100 Potenza, Italy
Database programming languages. International workshop, 6 (Estes Park CO USA) 1997-08-18
Journal: Lecture notes in computer science, **1998**, 1369 118-135
ISBN: 3-540-64823-2 ISSN: 0302-9743 Availability: INIST-16343; 354000076409400080
No. of Refs.: 29 ref.
Document Type: P (Serial); C (Conference Proceedings) ; A (Analytic)
Country of Publication: Germany; United States
Language: English
This paper develops a database query language called Transducer Datalog motivated by the needs of a new and emerging class of database applications. In these applications, such as text databases and genome databases, the storage and manipulation of long character sequences is a crucial feature. The issues involved in managing this kind of data are not addressed by traditional database systems, either in theory or in practice. To address these issues, in recent work, we introduced a new machine model called a generalized sequence transducer. These generalized transducers extend ordinary transducers by allowing them to invoke other transducers as " **subroutines** ." This paper establishes the computational properties of Transducer Datalog, a query language based on this new machine model. In the process, we develop a hierarchy of time-complexity classes based on the Ackermann function. The lower levels of this hierarchy correspond to well-known complexity classes, such as polynomial time and hyper-exponential time. We establish a tight relationship between levels in this hierarchy and the depth of **subroutine** calls within Transducer Datalog programs. Finally, we show that Transducer Datalog programs of arbitrary depth express exactly the sequence functions computable in primitive- **recursive** time.

?

33/7/3 **(Item 3 from file: 2)**
DIALOG(R)File 2:INSPEC

07164276 INSPEC Abstract Number: C1999-03-6160D-019
Title: A pattern of inheritance and polymorphism for persistent objects stored in a relational database
Author(s): Chung-Yeung Pang
Author Affiliation: Seveco AG, Buttwil, Switzerland
Journal: JOOP vol.11, no.9 p.41-4
Publisher: SIGS Publications,
Publication Date: Feb. 1999 Country of Publication: USA
CODEN: JOOPEC ISSN: 0896-8438
SICI: 0896-8438(199902)11:9L.41:PIPP;1-Y
Material Identity Number: G316-1999-002
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Because relational **databases** (RDBs) do not support object-oriented features like inheritance and polymorphism, persistent objects whose data are stored in an **RDB** cannot usually make full use of these features. A solution to this problem is presented in this article in the form of a design and C++ implementation pattern. In this pattern, the mapping of the persistent objects to **RDB** tables is addressed. A surrogate concept is used for the design of the persistent objects. The use of surrogates allows persistent objects to be extended through inheritance. Polymorphism is supported even though the structure of the persistent **objects** is **fixed** according to their **storage** in the **RDB** tables. (2 Refs)
Subfile: C

33/7/4 **(Item 4 from file: 2)**
DIALOG(R)File 2:INSPEC

07161818 INSPEC Abstract Number: C1999-03-6160D-018
Title: A framework for knowledge discovery from large relational databases
Author(s): Ajaja, M.K.; Abdel-Wahhab, A.H.; Shaheen, S.I.
Conference Title: Proceedings of the Fourth IEEE International Conference on Electronics, Circuits and Systems Part vol.2 p.635-9 vol.2
Publisher: Electron. Res. Inst, Cairo, Egypt
Publication Date: 1997 Country of Publication: Egypt 3 vol. xxxvii+1523 pp.
Material Identity Number: XX-1998-02933
Conference Title: Proceedings of 4th International Conference on Electronics Circuits and Systems (ICECS'97)
Conference Sponsor: Minstr. Int. Cooperation; IEEE CAS Soc.; IEEE Chapter
Conference Date: 15-18 Dec. 1997 Conference Location: Cairo, Egypt
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: Many researchers have proposed methods and operators for discovering different kinds of patterns from **databases** , like for example association rules, characteristic rules, and decision trees. Most of them assume that data is stored in a flat relational table, where each of the tuples corresponds to one **object** and has several properties **stored** in a **fixed** number of attributes. This format is not suitable for many real world applications and general discovery techniques. In this paper, a new framework is proposed that can extract patterns from data stored in normalized tables in relational **databases** . This framework uses mutual implication rules with forward and backward certainty measures and maps the patterns as a set of mutual implication rules. The discovery task is directed by a meta-knowledge query that directs the search for the required patterns. (16 Refs)
Subfile: C

33/7/6 (Item 6 from file: 2)
DIALOG(R)File 2:INSPEC

05401942 INSPEC Abstract Number: C9306-6160J-006
Title: Supporting physical independence in object databases
Author(s): Aloia, N.; Barneva, S.; Rabitti, F.
Author Affiliation: CNUCE-CNR, Pisa, Italy
Journal: Database Technology vol.4, no.4 p.265-86
Publication Date: 1991-1992 Country of Publication: UK
CODEN: DATEEA ISSN: 0951-9327
U.S. Copyright Clearance Center Code: 0951-9327/92/$5.00+0.00
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: In the current implementations of object **database** systems, the physical **database** organization is usually hard-coded in the system, i.e. the strategy for **memorizing** the data **objects** is **fixed** and cannot be changed by the user. The physical object organization usually reflects only the logical object definitions (i.e. the class definition). The authors provide the **database** designer with capabilities which permit him to choose the most suitable physical organization for an object **database** , in order to meet the performance requirements of particular applications. For this purpose, they have defined a canonical object data model and a storage object data model. In the first, the objects are organized in classes and basic operations on classes and objects are provided. In the second (which is similar to the data media control language of traditional **database** systems), physical objects with similar structures are grouped in collections. Operations for grouping and partitioning logical objects into physical objects are provided. As a result, a collection may correspond to one or more classes (or parts of them) of the canonical **database** schema. The languages used to describe the canonical object data model and the storage object data model are presented and the mechanisms for mapping data structures and operations from one level to the other are discussed. (14 Refs)
Subfile: C